UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/731,515	12/09/2003	Marcin Sawicki	60001.295US01	3732

27488 7590 08/02/2007

MERCHANT & GOULD (MICROSOFT)
P.O. BOX 2903
MINNEAPOLIS, MN 55402-0903

EXAMINER	
RIES, LAURIE ANNE	
ART UNIT	PAPER NUMBER
2176	
MAIL DATE	DELIVERY MODE
08/02/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/731,515	SAWICKI ET AL.	
	Examiner	Art Unit	
	Laurie Ries	2176	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *05 June 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1 and 3-23* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3,4,6-15 and 17-23* is/are rejected.

7)☒ Claim(s) *5 and 16* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *09 December 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to communications: Amendment, filed 5 June 2007, to the Original Application, filed 9 December 2003.

2. Claims 1, and 3-23 are pending. Claims 1, 11, and 19 are independent claims.

Response to Arguments

3. Applicant's arguments filed 5 June 1007 have been fully considered but they are not persuasive.

Applicant argues that Ayers fails to teach an application document that has been generated by an application that uses a file format that is specific to the application. The Office respectfully disagrees. Ayers teaches that a file is stored in .abw format, which is native to AbiWord. Specifically, Ayers teaches: "The most significant difference between AbiWord and nearly every other word processor available is the nature of the native file format (See Ayers, Page 2, fourth paragraph).

Applicant further argues that Ayers fails to teach mapping the determined properties of the list into at least one of a markup language element...and storing the mapped properties of the list in the markup language document. The Office respectfully disagrees. Ayers teaches that the code is the markup language mapped from the

properties of the document field displayed in the screenshot shown on Ayers, page 3. Ayers further teaches saving the document shown in the screenshot on Page 3, which creates the saved code that is saved in memory.

It is also noted that in the previous Office action, filed 5 March 2007, the entirety of claims 5 and 16, as written in the response filed 26 December 2006, were listed as allowable subject matter. Amending the independent claims to include only a portion of dependent claims 5 and 16 does not render the independent claims allowable and, additionally, changes the scope of the independent claims.

Claim Rejections - 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claims 3-4 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claims 3 and 4 are dependent upon claim 2, which has been cancelled. For the sake of further prosecution, it is assumed that claims 3 and 4 should be dependent upon claim 1.

Claims Rejection – 35 U.S.C. 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1, 3-4, 6-15, and 17-23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ayers, "AbiWord's Potential", hereinafter "Ayers", in view of Rohr, "RE: Styles Again", hereinafter "Rohr", W3C, "XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001, hereinafter "XML Schema", W3C, "XML Schema Requirements, hereinafter "XML Requirements", and Vermeire (U.S. Patent 6,209,124 B1).

As per independent claim 1 and dependent claim 6, Ayers teaches

A method for representing field structures in a markup language document, comprising:

inputting an application document that has been generated by an application that uses a file format that is specific to the application.

Specifically, the specification discloses that the invention "provides a method to represent an application's native field structures in markup language (ML) such as XML." See, Disclosure, page 6, lines 12-13. Ayers teaches: "The most significant difference between AbiWord and nearly every other word processor available is the nature of the native file format. An *.abw file is written in XML and thus is also in ASCII

format; the files can be read by any text editor." See, Ayers, page 2, fourth paragraph. Therefore, Ayers teaches the limitation of "a document that has been generated by an application that uses a file format that is specific to the application," and more specifically, teaches a native file format in a markup language, specifically XML.

Ayers also teaches

determining properties corresponding to a field that relates to at least one section of an application document;

It is noted that the term "properties" is not specifically defined in the specification. The disclosure associates the term in a general sense with simple and complex fields, as in the following: "the properties of the complex field (when the field is a complex field) are mapped into elements, attributes, and values of the ML file." Disclosure, page 18, lines 9-10. "Attributes" are also noted as one of the properties elsewhere in the disclosure. See, disclosure, page 3, line 26. The disclosure also defines that an element may have no attribute associated with it. See, disclosure, page 3, lines 27-28. Further evidence that the term refers to the general appearance of the document is taken from the disclosure, stating: "the fields and the properties associated with the fields may change from page to page, section to section, chapter to chapter and the like." Disclosure, page 18, lines 22-23. Based on the above analysis, it is believed that the applicants intended the term "properties" to be used in the general sense of the appearance of the text, and such definition will be used for the remainder of this Office Action. In support of the above definition of "properties" as known to one of ordinary skill in the art at the

time of the invention, see, Harold, Elliotte Rusty, "XML Bible," IDG Books Worldwide, 1999, pages 369-388.

Ayers also teaches

mapping the properties of the field into at least one of a markup language element, an attribute, and a value; and

See, Ayers, page 3, last two full paragraphs and citing code example from the screenshot and showing font and line properties and values. Note that Ayers teaches that the code is the markup language mapped from the properties of the document field displayed in the screenshot.

Ayers also teaches

storing the properties of the field in the markup language document, whereby applications different from the application can understand the mapped list properties stored in the markup language document..

(See, Ayers, page 3, last two full paragraphs and citing code example from the screenshot. Note that Ayers teaches saving the document shown in the screenshot, which creates the saved code that is saved in memory.

Ayers also teaches:

Where the file format is in a format that is native to the application

Specifically, Ayers teaches that "The most significant difference between AbiWord and nearly every other word processor available is the nature of the native file format. An *.abw file is written in XML and thus is also in ASCII format; **the files can be read by any text editor**." See, Ayers, page 2, fourth paragraph, emphasis added.)

Ayers also teaches complex and simple fields.

(See, Ayers, page 3, screenshot and the last two full paragraphs, wherein the screenshot teaches both simple and complex fields, with simple and complex elements, and the code example teaches that the simple field is mapped and stored. Ayers teaches the screenshot with the text to be saved, and Ayers teaches code generated in XML from saving the text in AbiWord. Ayers does not expressly teach to determine whether the field is one of a complex field and a simple field.

XML Schema teaches that simple and complex data elements are defined as part of the XML language. See, XML Schema, downloaded page 7.

Ayers and XML Schema are analogous art because they are from the same field of endeavor of creating and manipulating electronic documents. At the time the invention was made, it would have been obvious to one of ordinary skill in the art to differentiate between a simple element and a complex element, and to use such differentiation to determine how to render a word document. It is at least obvious, and possibly inherent, from the screenshot and the code example shown in Ayers that AbiWord determines whether the data is complex or simple.

The suggestion or motivation for combining the teachings of Ayers, that a word processor program could convert and store documents in XML, with the teaching of XML Schema, including that document types are distinguished as simple or complex, is expressly stated in XML Requirements, stating: "The following usage scenarios describe XML applications that should benefit from XML schemas. * * * 4. Traditional

document authoring/editing governed by schema constraints." See, XML Requirements, page 3.

The combination of the teachings of Ayers to convert and store word processing documents in XML combined with the teachings of XML Schema that data may be one of two types, simple or complex, would be an invention whereby a document mapped to XML would also distinguish and map simple and complex elements, and that such could be stored in memory.)

Ayers also does not teach expressly that *the file format includes unique properties for describing fields within the document, where the unique properties are defined by the application.*

Rohr teaches including unique properties for describing fields within a document, such as character properties and paragraph properties. Rohr further teaches that the unique properties relate to at least one section of the application document, such as a paragraph within the application document (See Rohr, entire document).

Ayers and Rohr are analogous art because they are from the same field of endeavor of creating and manipulating electronic documents.

At the time of the invention it would have been obvious to one of ordinary skill in the art to include the definition of character and paragraph properties of Rohr with the functionality of Ayers. The motivation for doing so would have been to set the properties of the various components of the document, such as affecting the color of the text within a paragraph of the document (See Rohr, Page 1, italicized text at mid-page).

Ayers also does not teach expressly that a field is *designated with a simple field markup language element when the field is determined to be a simple field.* Vermeire teaches describing a simple data type or field within the <field> tag in an XML document.

Ayers, XML Schema, XML Requirements, Rohr, and Vermeire are analogous art because they are from the same field of endeavor of creating and manipulating electronic documents.

At the time of the invention it would have been obvious to one of ordinary skill in the art to include the designation of a simple field type identifier in an XML document of Vermeire with the XML document of Ayers, providing the benefit of allowing the user to determine the type of data being represented within the XML document such that it models the structure of most programming languages (See Vermeire, Column 8, lines 32-34).

As per independent claim 11, Ayers teaches:

A computer-readable medium for representing fields in a markup language document, comprising:

inputting an application document that has been generated by an application that uses a file format that is specific to the application;

(The specification discloses that the invention "provides a method to represent an application's native field structures in markup language (ML) such as XML." See, Disclosure, page 6, lines 12-13. Ayers teaches: "The most significant difference between AbiWord and nearly every other word processor available is the nature of the

native file format. An *.abw file is written in XML and thus is also in ASCII format; the files can be read by any text editor." See, Ayers, page 2, fourth paragraph. Therefore, Ayers teaches the limitation of "a document that has been generated by an application that uses a file format that is specific to the application," and more specifically, teaches a native file format in a markup language, specifically XML.)

Ayers also teaches:

> *determining properties relating to one or more fields used within the application document.*

It is noted that the term "properties" is not specifically defined in the specification. The disclosure associates the term in a general sense with simple and complex fields, as in the following: "the properties of the complex field (when the field is a complex field) are mapped into elements, attributes, and values of the ML file." Disclosure, page 18, lines 9-10. "Attributes" are also noted as one of the properties elsewhere in the disclosure. See, disclosure, page 3, line 26. Further evidence that the term refers to the general appearance of the document is taken from the disclosure, stating: "the fields and the properties associated with the fields may change from page to page, section to section, chapter to chapter and the like." Disclosure, page 18, lines 22-23. Based on the above analysis, it is believed that the applicants intended the term "properties" to be used in the general sense of the appearance of the text, and such definition will be used for the remainder of this Office Action. In support of the above definition of "properties" as known to one of ordinary skill in the art at the time of the invention, see, Harold, Elliotte Rusty, "XML Bible," IDG Books Worldwide, 1999, pages 369-388.

Ayers also teaches:

determining whether the field is one of a complex field and a simple field;

Specifically, See, Ayers, page 3, screenshot and the last two full paragraphs, wherein the screenshot teaches both simple and complex fields, with simple and complex elements, and the code example teaches that the simple field is mapped and stored. Ayers teaches the screenshot with the text to be saved, and Ayers teaches code generated in XML from saving the text in AbiWord.

Ayers does not expressly teach to determine whether the field is one of a complex field and a simple field.

XML Schema teaches that simple and complex data elements are defined as part of the XML language. See, XML Schema, downloaded page 7.

Ayers and XML Schema are analogous art because they are from the same field of endeavor of creating and manipulating electronic documents. At the time the invention was made, it would have been obvious to one of ordinary skill in the art to differentiate between a simple element and a complex element, and to use such differentiation to determine how to render a word document. It is at least obvious, and possibly inherent, from the screenshot and the code example shown in Ayers that AbiWord determines whether the data is complex or simple.

The suggestion or motivation for combining the teachings of Ayers, that a word processor program could convert and store documents in XML, with the teaching of XML Schema, including that document types are distinguished as simple or complex, is expressly stated in XML Requirements, stating: "The following usage scenarios

describe XML applications that should benefit from XML schemas. * * * 4. Traditional document authoring/editing governed by schema constraints." See, XML Requirements, page 3.

The combination of the teachings of Ayers to convert and store word processing documents in XML combined with the teachings of XML Schema that data may be one of two types, simple or complex, would be an invention whereby a document mapped to XML would also distinguish and map simple and complex elements, and that such could be stored in memory.)

Ayers also teaches:

writing the properties into at least one of a markup language element, an attribute, and a value; and

Specifically, See, Ayers, page 3, last two full paragraphs and citing code example from the screenshot and showing font and line properties and values. Note that Ayers teaches that the code is the markup language mapped from the properties of the document field displayed in the screenshot.

Ayers also teaches:

storing the properties in the markup language document such that the fields of the application document are substantially maintained when the markup language document is parsed by an application that is different from the application used to generate the application document.

Specifically, See, Ayers, page 3, last two full paragraphs and citing code example from the screenshot. Note that Ayers teaches saving the document shown in the screenshot, which creates the saved code that is saved in memory.

Ayers also teaches:

Where the file format is in a format that is native to the application.

Specifically, Ayers teaches: "The most significant difference between AbiWord and nearly every other word processor available is the nature of the native file format. An *.abw file is written in XML and thus is also in ASCII format; **the files can be read by any text editor**." See, Ayers, page 2, fourth paragraph, emphasis added.)

Ayers, XML Schema and XML Requirements do not teach expressly that *the file format includes unique properties for describing fields within the document, where the unique properties are defined by the application.*

Rohr teaches including unique properties for describing fields within a document, such as character properties and paragraph properties. Rohr further teaches that the unique properties relate to at least one section of the application document, such as a paragraph within the application document (See Rohr, entire document).

Ayers also does not teach expressly that a field is *designated with a simple field markup language element when the field is determined to be a simple field.*

Vermeire teaches describing a simple data type or field within the <field> tag in an XML document.

Ayers, XML Schema, XML Requirements, Vermeire, and Rohr are analogous art because they are from the same field of endeavor of creating and manipulating electronic documents.

At the time of the invention it would have been obvious to one of ordinary skill in the art to include the designation of a simple field type identifier in an XML document of Vermeire with the XML document of Ayers, providing the benefit of allowing the user to determine the type of data being represented within the XML document such that it models the structure of most programming languages (See Vermeire, Column 8, lines 32-34).

At the time of the invention it would have been obvious to one of ordinary skill in the art to include the definition of character and paragraph properties of Rohr with the functionality of Ayers, XML Schema and XML Requirements. The motivation for doing so would have been to set the properties of the various components of the document, such as affecting the color of the text within a paragraph of the document (See Rohr, Page 1, italicized text at mid-page).

As per independent claim 19, Ayers and Rohr in view of XML Schema and further in view of XML Requirements teaches:

A system for representing fields in a markup language document, comprising:

an application that is configured to:

input an application document that has been generated by an application that uses a file format that is specific to the application;

determine properties relating to a field included in at least one section of an application document;

determine whether the field is one of a complex field and a simple field;

map the properties into at least one of a markup language element, an attribute, and a value; and

store the properties in the markup language document; and

a validation engine configured to validate the markup language document.

It is noted that claim 19 incorporates substantially similar subject matter as claimed in claim 11, and in further view of the following, is rejected along the same rationale.

Validation of an XML document is expressly taught in XML Schema, which states: "An instance document may be processed against a schema to verify whether the rules specified in the schema are honored in the instance. Typically, such processing actually does two things, (1) it checks for conformance to the rules, a process called schema validation . . .." See, XML Schema, downloaded page 59. See also, XML Schema, downloaded page 60, expressly teaching validation of simple and complex element types. It would have been obvious to one of ordinary skill in the art at the time of the invention to validate an instance document in order to confirm that it follows the rules of the schema.

The specification discloses that the invention "provides a method to represent an application's native field structures in markup language (ML) such as XML." See, Disclosure, page 6, lines 12-13. Ayers teaches: "The most significant difference

between AbiWord and nearly every other word processor available is the nature of the native file format. An *.abw file is written in XML and thus is also in ASCII format; the files can be read by any text editor." See, Ayers, page 2, fourth paragraph. Therefore, Ayers teaches the limitation of "a document that has been generated by an application that uses a file format that is specific to the application," and more specifically, teaches a native file format in a markup language, specifically XML.

As per dependent claim 3, Ayers, Rohr, Vermeire, XML Schema, and XML Requirements teach:

> *The method of Claim 2, wherein an instruction portion of the field comprises at least one of richly formatted text and embedded additional fields when the field is a complex field.*

See Ayers, page 2, third full paragraph, teaching that AbiWord can be saved in RTF format.

As per dependent claim 4, Ayers, Rohr, Vermeire, XML Schema, and XML Requirements teach:

> *wherein an instruction portion of the field excludes richly formatted text and embedded additional fields when the field is a simple field.*

It is noted that the term "richly formatted text" is not defined in the application. It is believed that applicants intended to refer to Microsoft Corporation's Rich Text Format (RTF) standard, and this definition will be used for the remainder of this Office Action.

It is inherent in the XML schema that simple types cannot have element content and cannot carry attributes, and it is therefore inherent in AbiWord, as taught by Ayers

to save documents in XML. See in support of this inherent property, XML Schema, downloaded page 7.

As per dependent claim 7, Ayers, Rohr, Vermeire, XML Schema, and XML Requirements teach the limitations of claims 1 as described above. Ayers also teaches:

The method of Claim 1, further comprising:

determining properties corresponding to an additional field that relates to at least one section of the application document;

mapping the properties of the additional field into at least one of a markup language element, an attribute, and a value; and

storing the properties of the additional field in the markup language document.

It is noted that this claim is read as the method including to edit and add or insert text to a document and then to convert to XML and save the resulting combined document.

(See, Ayers, page 3, last two lines, teaching the editing, or modification, of an XML document in AbiWord.)

As per dependent claim 8, Ayers, Rohr, Vermeire, XML Schema, and XML Requirements teach the limitations of claims 1 as described above. Claim 8 incorporates substantially similar subject matter as claimed in claim 1, and in further view of the following, is rejected along the same rationale.

It is noted that claim 1 specifies storing "the field," which is a singular "field." See, claim 1, lines, 5 and 9. There may be many fields in a document. Claim 8 further limits claim 1 by specifying a repetition of the field storage process until "all fields" have

been stored. Claim 8 merely specifies completing the field storage process, which is the equivalent of storing the document.

Ayers teaches the storage of the document, which teaches the storage of all fields in the document. See, Ayers, page 3, last two full paragraphs and citing code example from the screenshot. Note that Ayers teaches saving the document shown in the screenshot, which creates the saved code that is saved in memory.

As per dependent claim 9, Ayers, Rohr, Vermeire, XML Schema, and XML Requirements teach the limitations of claims 1 as described above. Ayers also teaches:

The method of Claim 1, wherein the properties of the fields stored in the markup language document are understood by an application that understands the markup language when the field is not native to the application.

Specifically, See, Ayers, page 2, third full paragraph, teaching that an AbiWord document is saved in an *.abw file written in XML and that the files can be read by any text editor. IN addition, Ayers teaches fields stored in XML that are understood by an application that understands the markup language when the field is not native to the application. See, Ayers, page 2, fourth paragraph teaching, as follows: "AbiWord can also save in the HTML and RTF formats, both of which are accessible with word processors such as MS-Word and WordPerfect." Therefore, AbiWord's fields in XML are understood by markup languages not native to AbiWord, such as HTML.

As per dependent claim 10, Ayers, Rohr, Vermeire, XML Schema, and XML Requirements teach the limitations of claims 1 as described above. Ayers also teaches:

The method of Claim 1, wherein the markup language document is manipulated on a server to substantially reproduce the field of the application document notwithstanding the presence of an application that generated the markup language document.

Specifically, See, Ayers, page 2, third full paragraph, teaching that an AbiWord document is saved in an *.abw file written in XML and that the files can be read by any text editor. See also, Ayers, page 3, bottom two lines, teaching that the document save in AbiWord may be modified by hand in an editor rather than a word processor. It is inherent from the ability to be read by any text editor and be modified by hand that the document may be manipulated on any suitable computing device, including a server.

As per dependent claim 12, claim 12 incorporates substantially similar subject matter as claimed in claim 9, and is rejected along the same rationale.

As per dependent claim 13, claim 13 incorporates substantially similar subject matter as claimed in claim 10, and is rejected along the same rationale.

As per dependent claim 14, claim 14 incorporates substantially similar subject matter as claimed in claim 3, and is rejected along the same rationale.

As per dependent claim 15, claim 15 incorporates substantially similar subject matter as claimed in claim 4, and is rejected along the same rationale.

As per dependent claim 17, Ayers, Rohr, Vermeire, XML Schema, and XML Requirements teach the limitations of claims 11 as described above.

Ayers also teaches:

The method of Claim 11, further comprising representing the field with at least one of a fldChar element and instrText element when the field is determined to be a complex field.

The rejection of claim 11 is incorporated herein by this reference.

It is noted that the ability to create an element type is inherent in the XML language and is therefore inherent in AbiWord, which is taught by Ayers to save documents in XML.

As per dependent claim 18, claim 18 incorporates substantially similar subject matter as claimed in claim 7, and is rejected along the same rationale.

As per dependent claim 20, claim 20 incorporates substantially similar subject matter as claimed in claim 9, and is rejected along the same rationale.

As per dependent claim 21, claim 21 incorporates substantially similar subject matter as claimed in claim 3, and is rejected along the same rationale.

As per dependent claim 22, claim 22 incorporates substantially similar subject matter as claimed in claim 4, and is rejected along the same rationale.

As per dependent claim 23, claim 23 incorporates substantially similar subject matter as claimed in claim 10, and is rejected along the same rationale.

Allowable Subject Matter

6. Claims 5 and 16 objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

Conclusion

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Laurie Ries whose telephone number is 571-272-4095. The examiner can normally be reached on M-F, 6:00am-3:30pm. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Doug Hutton, can be reached on 571-272-4137. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

8. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Laurie Ries
Patent Examiner
Art Unit 2176